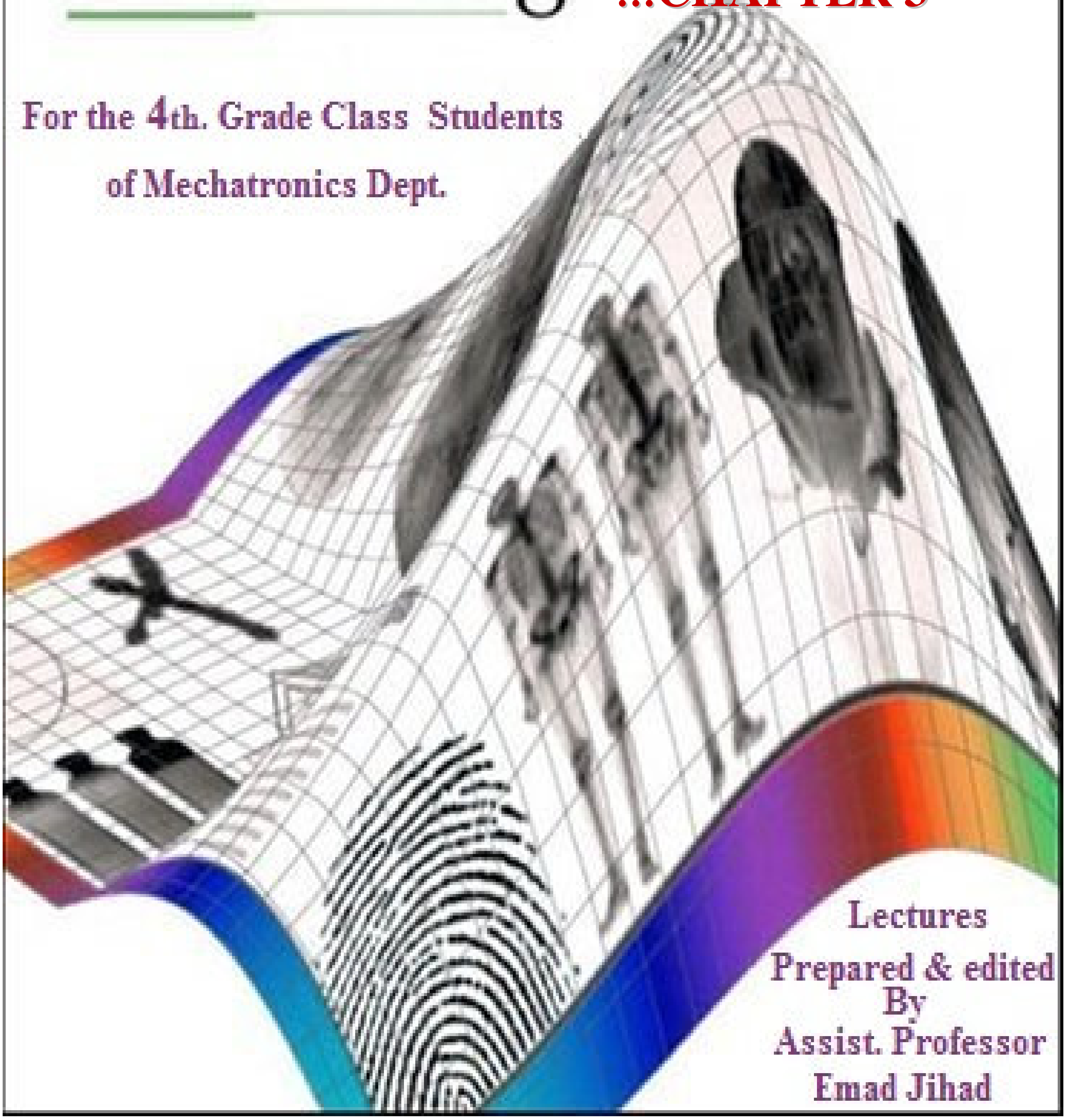
Digital Image Processing USING MATLAB

…CHAPTER 3

For the 4th. Grade Class Students

of Mechatronics Dept.



Lectures

Prepared & edited

By

Assist. Professor

Emad Jihad

3- Manipulating and Enhancing images

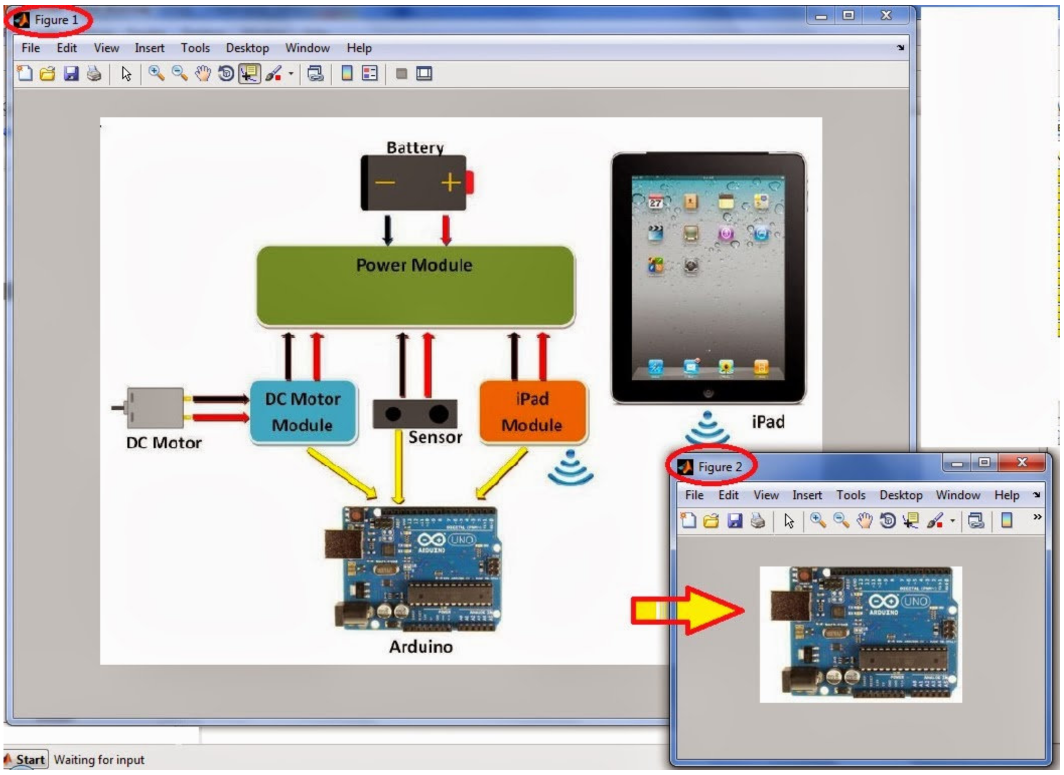
3-1 Sub imaging

In Chapter 2, item 2-4, we gave an example of image crop using image tool. Alternatively, we give here a programming script to do this procedure.

As an image is a 2 dim. Array of pixel values, thus we can get a sub image from the original one.
The new sub image will be defined as the following: IS = Iss (x1: x2, y1:y2, **1:3**)
Where (IS) is any name for the source image, (Iss) is the new sub image, x1, x2, y1, and y2 are the beginning and ending pixels as rows and columns, while (1:3) is the third dimension which gives the three RGB color channels used (i.e. R=1, G=2, and B=3).

Ex 3-1: Perform crop or sub image segmentation from a given source image?

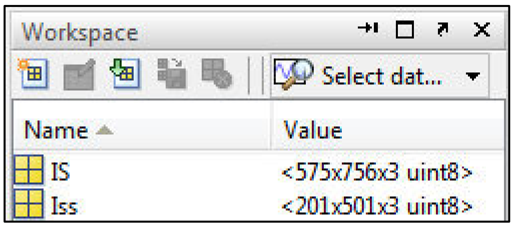
Write Matlab script to arbitrary Clip a segment (sub image) from a given source image as illustrated in the two figures below, and compare image sizes before and after. This procedure is to be repeated as long as user needs.
The size of source and the new segment image is to be displayed too.



Solution:

```
%****************************************
%* Program Script to arbitrary selecting Sub image *
%* From a source image by defining area to clip!    *
%****************************************

key=1;
while key ==1
   clc
   clear all
   IS=imread ('MechaTron.jpg');
   figure
   imshow (IS);
   input ('Enter x1: ');
   x1=ans;
   input ('Enter x2: ');
   x2=ans;
   input ('Enter y1: ');
   y1=ans;
   input ('Enter y2: ');
   y2=ans;
   Iss=IS(y1:y2,x1:x2,1:3);
   figure
   imshow(Iss);
   s1 = size(IS);
   s2 =size(Iss);
   display ('==============================')
   display ('Size of Original image '); s1
   display ('Size of Sub image '); s2
   input ('Hit Enter key to End or  input 1 to Continue!');
   key =ans;
   close all
end
close all
```

<u>Note</u>: In Matlab, We can find also the No. of Image rows and columns from The Workspace window...as shown below:



Name	Value
IS	<575x756x3 uint8>
Iss	<201x501x3 uint8>

3-2 Changing Image Brightness

All image formats (except logical or B/W images) Brightness (intensity) can be modified by changing the value of image pixels.
Since each pixel has its own numeric value that range between (0~255), then by adding or subtracting a suitable value with caution not to exceed the resulting pixel value the acceptable range of (0~255), will yield a new image intensity.
To increase or decrease the Brightness of an image, just us (+) for Brighter or (-) for Darker image!

Ex 3-2-1: Increase the Brightness of an image first, and then Decrease the original image brightness by the same value of (50).

Solution:

```
>> SS = imread ('Fruits.jpg');
>> figure (1); imshow (SS);
>> SSB=SS+50;
>> figure (2); imshow (SSB);
>> SSD=SS-50;
>> figure (3); imshow (SSD);
```

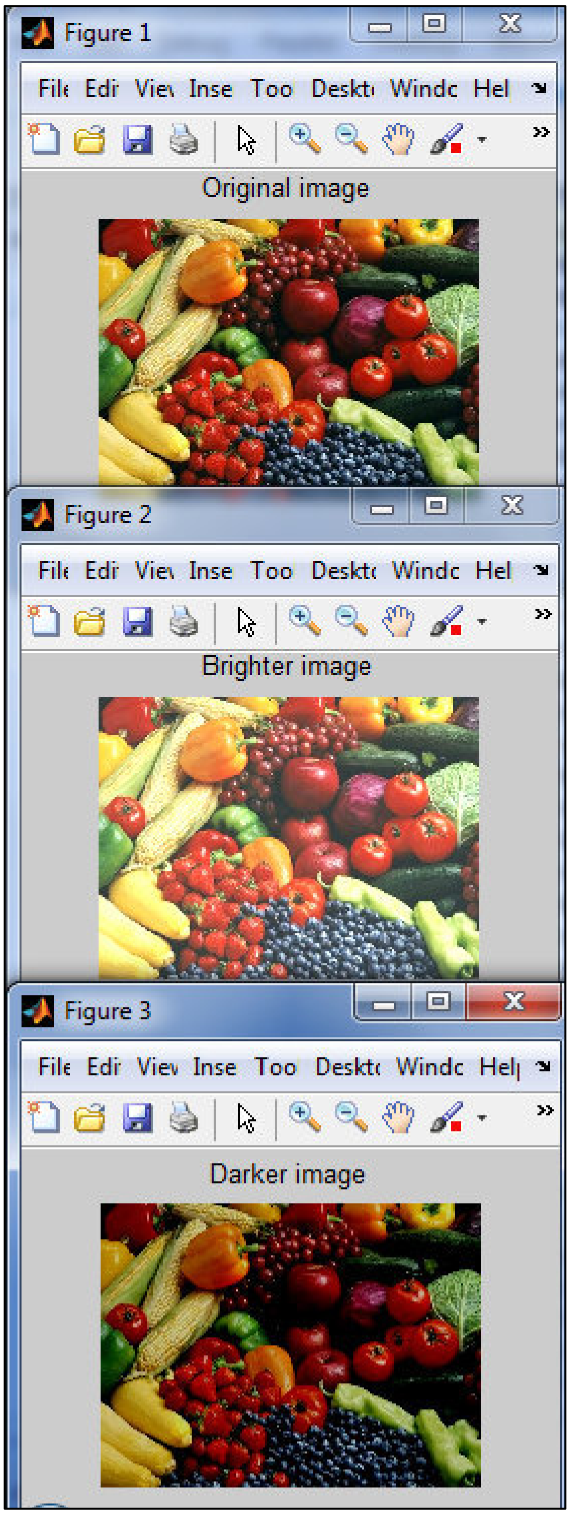


Figure 1
File Edit View Insert Tools Desktop Window Help
Original image
Figure 2
File Edit View Insert Tools Desktop Window Help
Brighter image
Figure 3
File Edit View Insert Tools Desktop Window Help
Darker image

Exercise: Write Matlab script to accomplish the Brightness modification by inputting modification value under loop control.

3-2-1 Changing Image Brightness for Gray level images

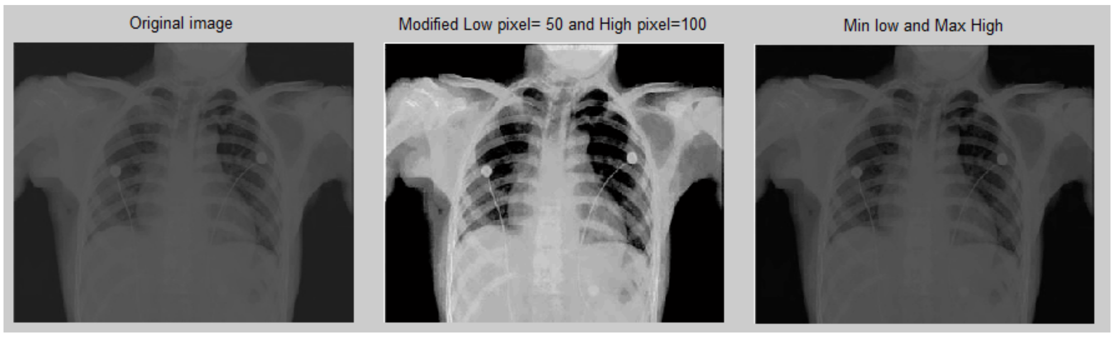
Using the syntax: **imread (f, [low, high]);**
Displays as Black all pixels values less or equal to the given value (low), and as White all values greater than or equal to (high).

While the syntax: **imread (f, [ ]);**
Sets variable low to the Minimum value of array f and high to its Maximum value. This special case of reading an image is useful for displaying images that have a low dynamic range or have negative and positive values.

Ex 3-2-1-1: Read a Gray level image then modify and display it with the new low and high pixels values and set those values to minimum and maximum values.

```
%**********************************************
%* Modifying the display of Gray level images!*
%* Program Name: ImageProEx08                 *
%**********************************************
clc
close all
f = imread ('xray1.jpg');
f = rgb2gray (f);
input ('Enter low  pixel value: '); low = ans;
input ('Enter High pixel value: '); high= ans;
subplot (131)
imshow (f)
title ('Original image')
subplot (132)
imshow (f,[low,high])
title (['Modified Low pixel= ',num2str(low),' and High 
pixel=',num2str(high)])
subplot (133)
imshow (f,[ ])
title ('Min low and Max High')
```

The output images after entering Low and High values will be:



Exercise: Try to show the images on separate figures (3 figures) to have better display

3-3 Complementing image pixels

IM2 = imcomplement (IM) computes the complement of the image IM. IM can be a binary, grayscale, or RGB image. IM2 has the same class and size as IM.

In the complement of a binary image, zeros become ones and ones become zeros; black and white are reversed. In the complement of an intensity or RGB image, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images) and the difference is used as the pixel value in the output image. In the output image, dark areas become lighter and light areas become darker.

This can turn Positive images to Negative and vise versa.

Ex 3-3-1: Write script to turn image file into Negative, or Negative to Positive image.

```
%*************************************************
%* Complementing image color (Negative effect)! *
%* But Works with other image types as well!    *
%* Program Name: ImageProEx11                   *
%*************************************************
clc
close all
fn = input ('Enter image file Name: ','s');
fn = imread (fn); % important step to have image matrix
imshow (fn)
title ('Original image')
input ('Hit Enter key to get Complemented image!');
figure
nfn = imcomplement (fn);
imshow (nfn)
title ('Complemented image')
```

For RGB color image:

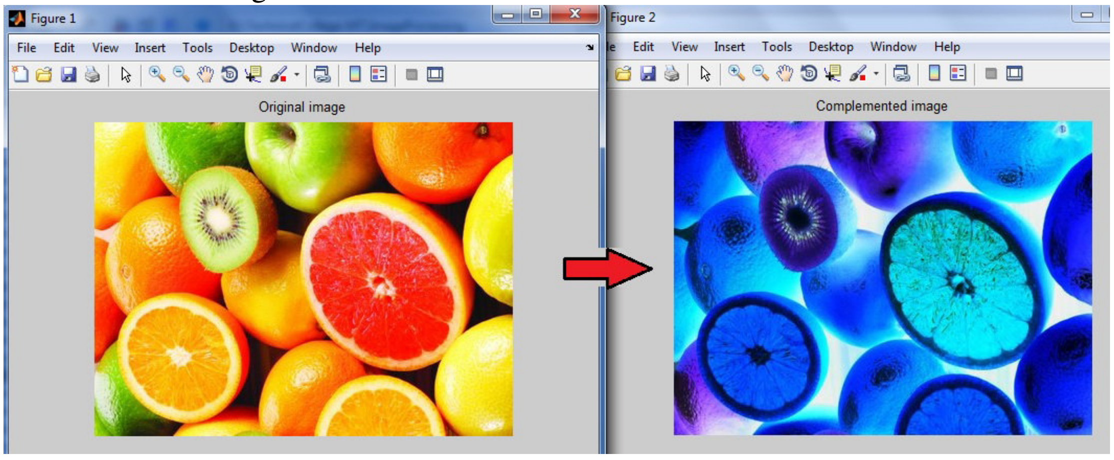
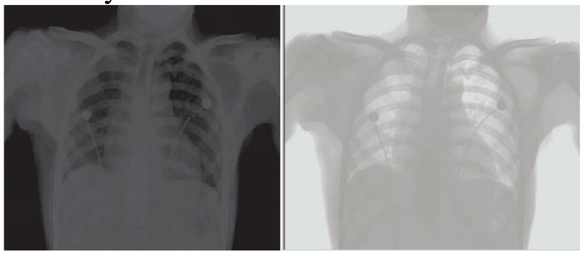
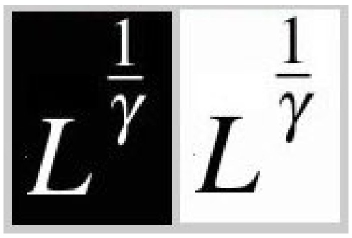


Figure 1
File Edit View Insert Tools Desktop Window Help
Original image
Figure 2
Edit View Insert Tools Desktop Window Help
Complemented image

For Gray scale .... and ....Binary (logical) images:





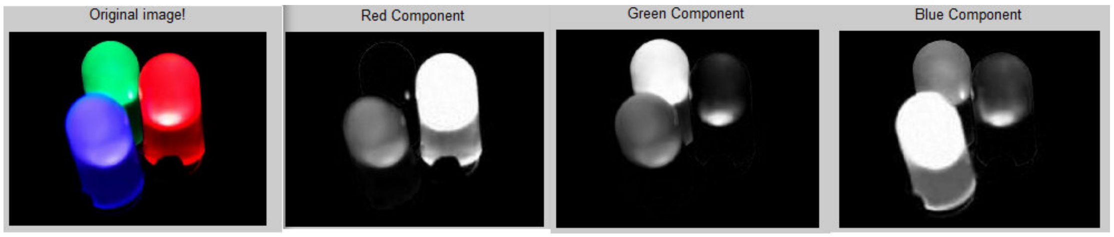
3-4-1 Red, Green, and Blue separate effect on images

As we have seen before, Color RGB images use in general three matrices for holding x, y intensity value in addition to the matrix No. which represents R, G, or B color.

Ex 3-4-1: The following script showing an example of R, G, and B effect on a color image:

```
%**********************************************************
%* Showing Red, Green, and Blue Components of color image!*
%* Program Name: ImageProEx14                             *
%**********************************************************
clc
im1 = input ('Enter Color image name: ','s');
im1 = imread(im1); % Reads an image and puts it in im1

figure; imshow(im1); title('Original image!');
figure; imshow(im1(:,:,1)); title('Red Component');
figure; imshow(im1(:,:,2)); title('Green Component');
figure; imshow(im1(:,:,3)); title('Blue Component');
input('Hit Enter key!');
figure;
subplot(4,1,1)
imshow(im1);
title('Original image!');
subplot(4,1,2)
imshow(im1(:,:,1)) % Displays Red component of the image
title('Red Component');
subplot(4,1,3)
imshow(im1(:,:,2)) % Displays Green component of image
title('Green Component');
subplot(4,1,4)
imshow(im1(:,:,3)) % Displays Blue component of the image
title('Blue Component');
truesize;
input('Hit Enter key to Close all!');
close all
```

One can see clearly the effect of R, G, and Blue on an image.
For instance, when Red is present in the absence of Green and Blue values, we notice that the Red LED in the second image below is very bright, while the Blue one is dimmed and the Green color is unseen, and so on!



3-4-2 Using Matlab built-in Colormap enhancement effects

When using indexed images, it's also become possible to use different enhancement techniques available. In Matlab there are built-in Colormap effects that can be used alone with such type of images.
Syntax:

colormap (map)
colormap ('default')
cmap = colormap
colormap (ax,...)

A colormap is an m-by-3 matrix of real numbers between 0.0 and 1.0. Each row is an RGB vector that defines one color. The kth row of the colormap defines the kth color, where map(k,:) = [r(k) g(k) b(k)]) specifies the intensity of red, green, and blue.

Colormap (map) sets the colormap to the matrix map. If any values in map are outside the interval [0 1], you receive the error Colormap must have values in [0,1].

colormap('default') sets the current colormap to the default colormap.

cmap = colormap retrieves the current colormap. The values returned are in the interval [0 1].

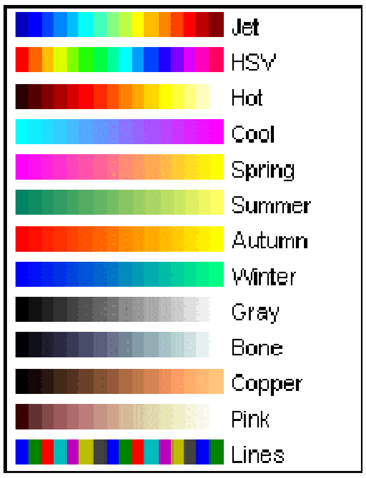
colormap(ax,...) uses the figure corresponding to axes ax instead of the current figure.

Specifying Colormaps

Files in the color folder generate a number of colormaps. Each file accepts the colormap size as an argument. For example,
colormap (hsv(128)) : creates an hsv colormap with 128 colors. If you do not specify a size, a colormap the same size as the current colormap is created.

Supported Colormaps

The built-in MATLAB colormaps are illustrated and described below. In addition to specifying **built-in** colormaps ***programmatically***, you can use the Colormap menu in the Figure Properties pane of the Plot Tools GUI to select one interactively.



3-5 Image Restoration

Image restoring is meaning the process of Retrieving the original image from degradation or after some effects that has taken place on it.

To restore the indexed mapped image that has some effect, we use the image conversion Matlab function:

```
RGB = ind2rgb(X,map);
```

This will return indexed image with an effect factor back to its original source image.

Comprehensive **Example** on using Matlab built-in effects and image Restoration:

Example 3-5-1: Write Matlab script to read 4 (jpg) files then convert them to indexed image type. Use built-in effects to enhance those images and show them.
Then select any of the indexed mapped images to restore back to the original type again!

Solution:

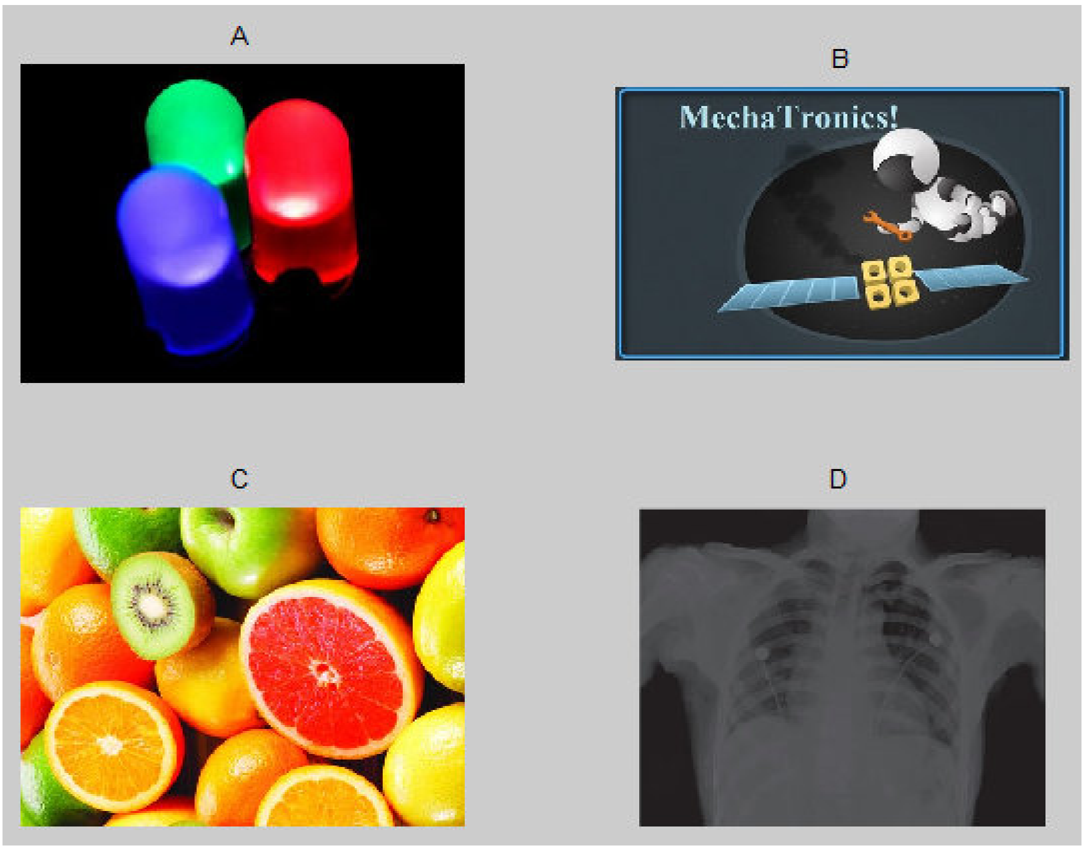
Program script can be as shown in the next pages, with all images out of processing.
It may be look a long program script, but actually this is because of using control to manage the processing in step-by-step easy to follow and understand procedures

```
%**************************************************
%* Using colormap to change indexed-image colors!*
%* Matlab built-in Colormaps are:                 *
%* jet, hsv, hot, cool, spring, summer, Autumn,   *
%* winter, gray, bone, copper, pink, and lines    *
%* Program Name: ImageProEx15                     *
%**************************************************
clc
clear all
A = imread ('RGBLED.jpg');
B = imread ('MechaT.jpg');
C = imread ('fruit-photo.jpg');
D = imread ('xray1.jpg');
subplot (221);
imshow (A);
title('A')
subplot (222);
imshow (B);
title('B')
subplot (223);
imshow (C);
title('C')
subplot (224);
imshow (D);
title('D')
truesize; display ('Images will be Changed into indexed images!')
input ('Hit Enter key to Continue!')
figure(2)
subplot (221);
RGB1 = A;
[X,map] = rgb2ind(RGB1,128);
imshow (X,map);

subplot (222);
RGB2 = B;
[X,map] = rgb2ind(RGB2,128);
imshow (X,map);

subplot (223);
RGB3 = C;
[X,map] = rgb2ind(RGB3,128);
imshow (X,map);

subplot (224);
RGB4 = D;
[X,map] = rgb2ind(RGB4,128);
imshow (X,map);

truesize;
display
('=============================================================')
display ('Now choose Colormap method No. to convert to:  ')
display ('1-Jet 2-HSV 3-Hot 4-Cool 5-Spring 6- Summer 7-Autumn')
display ('8-Winter 9-Gray 10-Bone 11-Copper 12-Pink 13-Lines!')
display
('=============================================================')
```

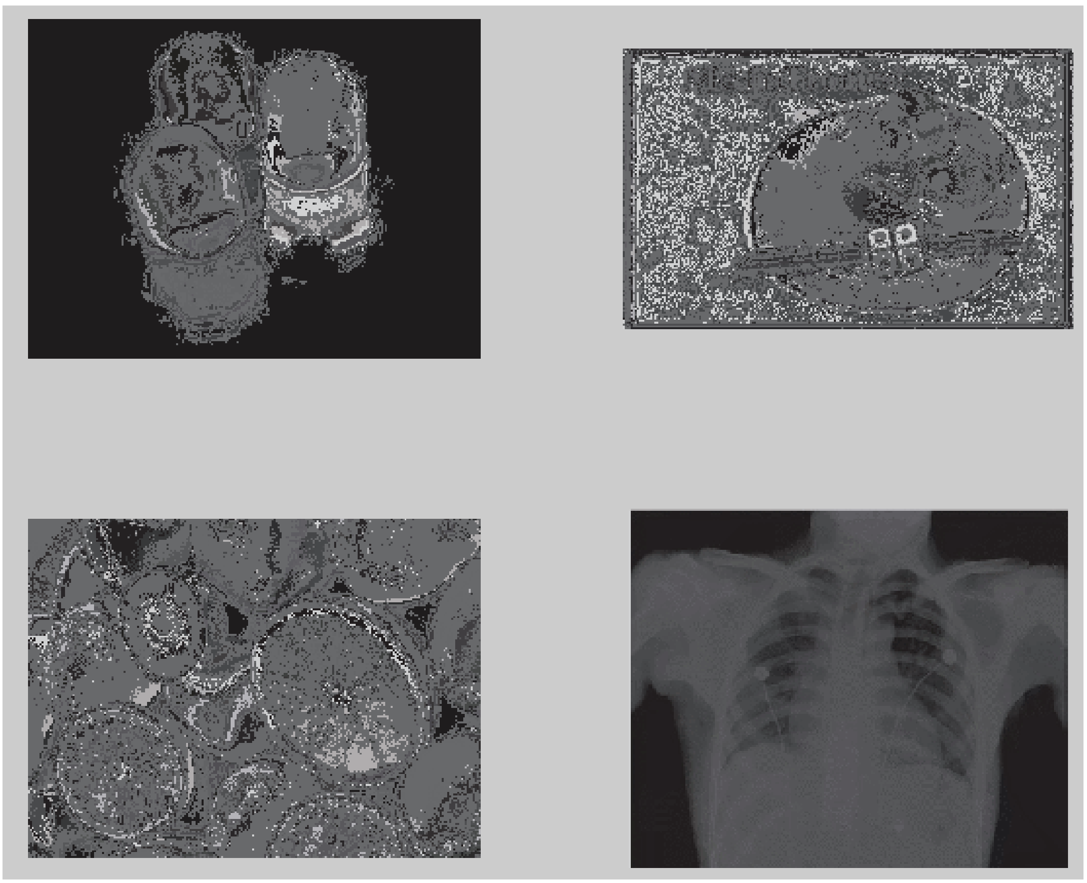
To be continued on the Next Page!

```
s = 1;
while (s < 14) && (s ~= 0)
    s = input ('Enter method No. (0 to Exit) : ');
    switch s

      case 1
         colormap(jet)
      case 2
         colormap(hsv)
      case 3
         colormap(hot)
      case 4
         colormap(cool)
      case 5
         colormap(spring)
      case 6
         colormap(summer)
      case 7
         colormap(autumn)
      case 8
         colormap(winter)
      case 9
         colormap(gray)
      case 10
         colormap(bone)
      case 11
         colormap(copper)
      case 12
         colormap(pink)
      case 13
         colormap(lines)
      otherwise
         disp('Unknown method!')
end % of switch statement
end % of while  statement
display
('================================================================')
ch = input ('Enter indexed mapped image No. to Restore back to
origin: ');
figure;
RGB = ind2rgb(X,map);
 switch ch
   case 1
      imshow (RGB1)
      title ('Original Image Restored!')
   case 2
      imshow (RGB2)
      title ('Original Image Restored!')
   case 3
      imshow (RGB3)
        title ('Original Image Restored!')
   case 4
      imshow (RGB4)
        title ('Original Image Restored!')
   otherwise
end

input ('Hit Enter key to Close figures!!!');
close all
```

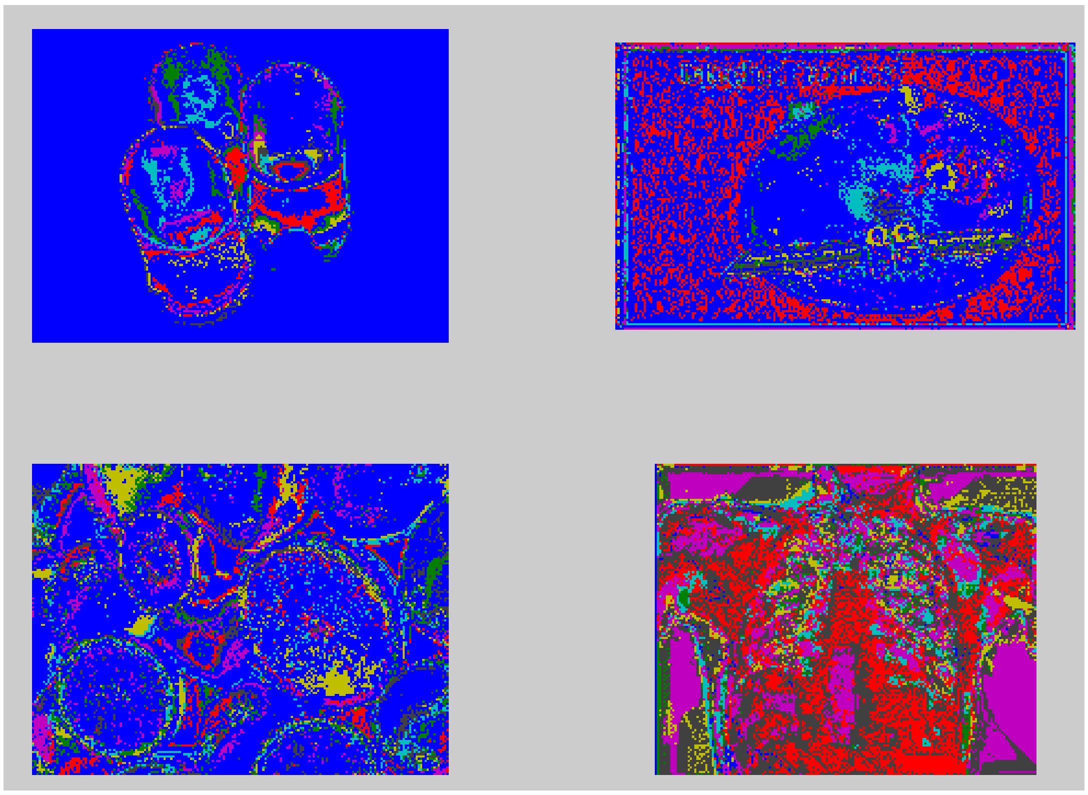
Original images together shown below:



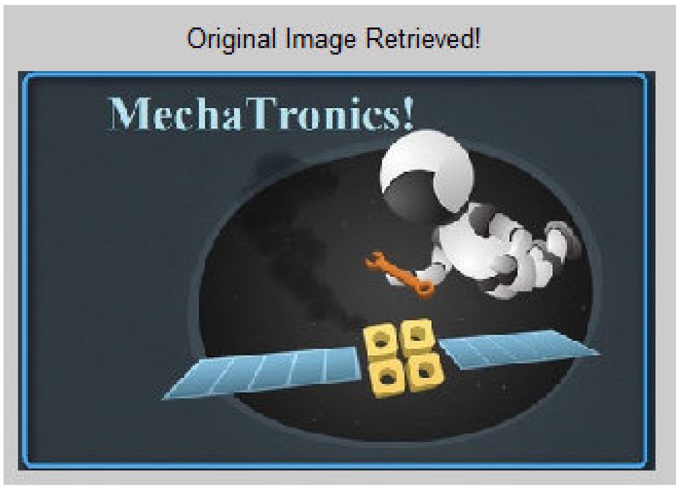
Images after being mapped to indexed images:



The images after enhanced using (lines) colormap effect (No. 13):



Choosing an image to be restored back to its original source, in this case the 2nd. image selected to be restored (Retrieved) as seen below:


Original Image Retrieved!
MechaTronics!